

Remarks

Claims 1-5, 7-14, 16-17 and 19-20 remain in the application. Claims 6 and 18 were previously canceled without prejudice. Claim 15 is hereby canceled without prejudice. Claims 1, 7, 12, 19, and 20 are hereby amended. No new matter is being added.

Claim Objections

Claims 7-9 were objected to for depending on canceled claim 6. Accordingly, applicants have hereby amended claim 7 so as to depend from claim 1. Claims 8-9 depend from claim 7. Hence, applicants respectfully submit that this objection is now overcome.

Claim Rejections--35 USC 103

Claims 1-5, 7-14, 16-17 and 19-20 were rejected under 35 U.S.C. 103 as being unpatentable over Quach in view of Fruehling (USP 6,625,688). This rejection is respectfully traversed.

Claims 1-5 and 7-11

Amended claim 1 now recites as follows.

1. A method of providing opportunistic functional testing within a central processing unit (CPU), the method comprising:
 - executing a computer program having both pre-scheduled redundant and non-redundant operations on multiple functional units of a same type in the CPU;
 - automatically comparing outputs from the multiple functional units; and
 - checking results of the comparison only for the pre-scheduled redundant operations but not for the pre-scheduled non-redundant operations, wherein the pre-scheduled redundant operations are opportunistically scheduled by a compiler during compilation of source code for the computer program and before beginning execution of the computer program to

execute redundantly on at least two of the multiple functional units to utilize an otherwise idle cycle for at least one of the multiple functional units for purposes of functional testing, and wherein multiple of the pre-scheduled non-redundant operations are scheduled by the compiler during compilation of the source code for the computer program and before beginning execution of the computer program to execute in parallel on the multiple functional units for purposes of performance.

(Emphasis added.)

As recited above, claim 1 now requires that “the pre-scheduled **redundant** operations are opportunistically scheduled by a compiler **during compilation of source code for the computer program and before beginning execution of the computer program** to execute redundantly on at least two of the multiple functional units to utilize an otherwise idle cycle for at least one of the multiple functional units for purposes of **functional testing**.” Claim 1 also requires that “multiple of the pre-scheduled **non-redundant** operations are scheduled by the compiler **during compilation of the source code for the computer program and before execution of the computer program** to execute in parallel on the multiple functional units for purposes of **performance**.” Claim 1 further requires “automatically comparing outputs from the multiple functional units” and “checking results of the comparison only for the pre-scheduled redundant operations but not for the pre-scheduled non-redundant operations.”

Claim 1 find support, for example, in the description from page 5, line 17 to page 7, line 14. For convenience of reference, this portion of the description is provided below.

FIG. 4a is a flow chart depicting steps relating to scheduling instructions by a compiler in accordance with an embodiment of the invention. The method of FIG. 4a may be utilized in a microprocessor or central processing unit (CPU) with multiple functional units of the same type. For example, the CPU may have multiple arithmetic logic units (ALUs) or multiple floating point units (FPUs).

Conventionally, it is a function of the compiler scheduler to keep all of these units as busy as possible. Nevertheless, there will be cycles when a unit will be idle or perform a no-op (no operation).

In accordance with an embodiment of the invention, per the method 60 of FIG. 4a, identification 62 is made of such a cycle in which a functional unit would be idle. Instead of letting the unit be idle, the compiler schedules 64 a redundant operation into that idle unit to provide opportunistic fault checking of the function. If more fault checking is desired by the user, a 'slider' can be set in the compiler by the user to more aggressively use the hardware comparator. As an example, suppose the slider could be set to any value from 0 to 10. A value of '0' could mean to only schedule compares when functional units would otherwise be idle. A value of '10' could mean to use all idle cycles, plus force all FPU / ALU instructions to be redundant. The higher the level of aggressiveness, the greater the level of fault checking provided (at the cost of performance). Other mechanisms may also be used to set the various levels of aggressiveness of the compiler.

For example, consider a CPU with two floating point units, FP_A and FP_B. Most of the time, both units may be scheduled to operate independently, and the comparison flag need not be checked. However, the compiler may be able to schedule an operation for FP_A for a given cycle, but there may not be another operation available for scheduling on FP_B for that cycle. In accordance with an embodiment of the invention, the compiler would identify 62 this opportunity and schedule 64 the same operation for both FP_A and FP_B.

FIG. 4b is a flow chart depicting steps during execution of a compiled program in accordance with an embodiment of the invention. Per the method 65 of FIG. 4b, operations are executed 66 by multiple functional units within the CPU. In accordance with an embodiment of the invention, these operations are scheduled by the compiler. Normally, different operations are scheduled for execution on the multiple functional units during a cycle. Occasionally, as described above, the intelligent scheduler opportunistically schedules an operation to be redundantly performed for purposes of fault checking.

The outputs from the multiple functional units are automatically compared 67 against each other by comparator circuitry within the target microprocessor. A comparison flag is set or reset 68 based on the resulting output of the comparator circuitry. This occurs for each cycle, regardless of whether different operations or redundant operations are executed on the multiple function units.

However, the comparison flag is to be checked and acted upon 69 only after the execution of such a redundant operation. The comparison flag would not be checked or acted upon after the execution of non-redundant operations because when different operations are performed by the multiple functional units, then the results are expected to be different. In other words, a difference in the results would not normally indicate an error by one of the functional units. However, when a redundant operation is executed on the multiple functional units, then the results are expected to be the same, unless there is an error by one of the functional units.

In the above example, the outputs of FP_A and FP_B may be configured to connect directly to a hardware comparator whose output will be "1" if the two results are the same and "0" if the results are different (or vice versa). During

normal operation, when FP_A and FP_B are operating on different data, the output of the comparator would be “0”, but this information will not be used. Only when a redundant operation is executed in both FP_A and FP_B will this flag be checked and acted upon if there is an error indicated by a “0” (due to the results being different). While this example cites floating point units, the principle is applicable to any processor resource of which there are multiple copies that are scheduled by the compiler.

As stated in the latest office action, “Quach fails to explicitly disclose a method wherein the redundant operations are opportunistically scheduled by a compiler to take advantage of an otherwise idle functional unit during a cycle.” The latest office action goes on to assert that “Fruehling clearly teaches a method of taking advantage of idle CPU cycles” and that it would have been obvious “to include the opportunistic behavior taught by Fruehling to take advantage of idle and unused processing cycles.”

However, applicants respectfully point out that neither Quach nor Fruehling teach **modifying a compiler** such that “the pre-scheduled **redundant** operations are opportunistically scheduled by a compiler during compilation of source code for the computer program and before beginning execution of the computer program to execute redundantly on at least two of the multiple functional units to utilize an otherwise idle cycle for at least one of the multiple functional units for purposes of **functional testing**” and such that “multiple of the pre-scheduled **non-redundant** operations are scheduled by the compiler during compilation of the source code for the computer program and before beginning execution of the computer program to execute in parallel on the multiple functional units for purposes of **performance**.”

Instead, both Quach and Fruehling teach methods pertaining to the operation of a microprocessor after compilation of the source code for the computer program and while executing the computer program. In particular, Quach teaches a “Microprocessor with High-Reliability Operating Mode,” and Fruehling teaches a “Method and Circuit for Analysis of the Operation of a Microcontroller using Signature Analysis of Memory.” (See titles of Quach and Fruehling.) Neither Quach nor Fruehling disclose or suggest methods pertaining to the operation of a compiler.

For at least the above-discussed reasons, applicants respectfully submit that amended claim 1 is now patentably distinguished over the cited art.

Claims 2-5 and 7-11 depend from claim 1. As such, applicants respectfully submit that claims 2-5 and 7-11 are now patentably distinguished over the cited art for at least the reasons discussed above in relation to claim 1.

Claims 12-14 and 16-17

For similar reasons as discussed above in relation to claim 1, applicants respectfully submit that claim 12 is also now patentably distinguished over the cited art.

Claims 13-14 and 16-17 depend from claim 12. As such, applicants respectfully submit that claims 13-14 and 16-17 are now patentably distinguished over the cited art for at least the reasons discussed above in relation to claim 12.

Claims 19 and 20

For similar reasons as discussed above in relation to claim 1, applicants respectfully submit that claims 19 and 20 are also now patentably distinguished over the cited art.

Conclusion

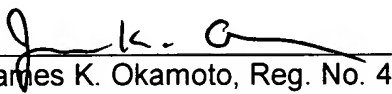
For the above-discussed reasons, applicant believes that claims 1-5, 7-14, 16-17 and 19-20, as they are hereby amended, are now patentably distinguished over the prior art. Favorable action is respectfully requested.

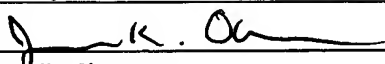
If for any reason an insufficient fee has been paid, the Commissioner is hereby authorized to charge the insufficiency to Deposit Account No. 08-2025.

Respectfully Submitted,

DALE JOHN SHIDLA et al.

Dated: January 22, 2007


 James K. Okamoto, Reg. No. 40,110
 Okamoto & Benedicto LLP
 P.O.Box 641330
 San Jose, CA 95164-1330
 Tel: (408) 436-2111
 Fax: (408) 436-2114

CERTIFICATE OF MAILING			
I hereby certify that this correspondence, including the enclosures identified herein, is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below. If the Express Mail Mailing Number is filled in below, then this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service pursuant to 37 CFR 1.10.			
Signature:			
Typed or Printed Name:	James K. Okamoto	Dated:	1/22/2007
Express Mail Mailing Number (optional):			